APRENDERAPROGRAMAR.COM

ADDEVENTLISTENER JAVASCRIPT. REMOVE. ATTACHEVENT. THIS EN EVENTOS. PROPAGACIÓN. BUBBLING. CAPTURA. (CU01158E)

Sección: Cursos
Categoría: Tutorial básico del programador web: JavaScript desde cero

Fecha revisión: 2029

Resumen: Entrega nº58 del Tutorial básico "JavaScript desde cero".

Autor: César Krall

ADDEVENTLISTENER JAVASCRIPT

Pueden ser objetivos (target) de eventos el objeto window, el objeto document y todos los nodos de tipo Element de un documento HTML. Todos estos objetos disponen de un método predefinido denominado addEventListener() que permite agregarles uno, dos o más manejadores de eventos.

La sintaxis a emplear para addEventListener es básicamente la siguiente:

```
nodoObjetivo.addEventListener ("nombreDeEventoSinPrefijo", funcionAEjecutar,
parametroBooleanoOpcional)
```

Donde nodoObjetivo será un nodo que habremos obtenido con un método como getElementById ó similar, nombreDeEventoSinPrefijo es el nombre del evento sin el prefijo on, por ejemplo click (en lugar de onclick como se escribiría tradicionalmente).

funcionAEjecutar es la función que se ejecutará cuando se produzca el evento.

Finalmente, el parámetroBooleanoOpcional especifica si el evento debe ser capturado (true) o no debe serlo (false, que es el valor que se aplica por defecto si no se especifica este parámetro. El tercer parámetro no lo especificaremos, o lo haremos como false. El motivo por el que se encuentra ahí es más histórico que una necesidad real y no vamos a detenernos a hablar sobre este parámetro.

Veamos un ejemplo:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">
window.onload = function () {
var elemsH = document.querySelectorAll("h1, h2, h3, h4, h5, h6");
for (var i=0;i<elemsH.length;i++) {
elemsH[i].addEventListener("mouseover", cambiarColor1);
elemsH[i].addEventListener("mouseout", cambiarColor2);}
function cambiarColor1() { this.style.color = 'orange';}
function cambiarColor2() { this.style.color = 'brown';}
}
</script>
</head>
<body><p><a href="principal.html" title="Página principal" >Ir a la pagina principal</a></p>
<h1>Novedades</h1><p>Aquí presentamos las novedades del sitio.</p>
<h3>Lanzamos el producto X-FASHION</h3>
<p>Este producto permite estirar la piel hasta dejarla como la de un bebé.</p>
```

```
<p><img src="http://i.imgur.com/tq5Bs.png" title="Imagen del producto" alt="imagen producto x-fashion" /></p>
<h3>Mejoramos el producto T-MOTION</h3>
<p>Hemos lanzado una nueva versión del producto T-MOTION</p>
<p><img src="http://i.imgur.com/SpZyc.png" title="Imagen del proudcto tmotion" alt="imagen producto tmotion" /></p>
</body>
</html>
```

El resultado esperado es que cuando se pase el mouse por encima de un elemento de título <h1>, <h2>, etc. el color cambie a naranja. Una vez el mouse deja de estar encima del elemento, el color del texto cambiará a marrón. Ten en cuenta que en algunos navegadores antiguos el código puede que no funcione.

PALABRA CLAVE THIS

Recordar que dentro de la función de respuesta a un evento, la palabra clave this hace referencia al elemento que es quien recibe (objetivo o target) el evento. En el ejemplo anterior vemos un ejemplo con el código this.style.color = 'orange'; que se encarga de cambiar el color del texto del elemento HTML donde se ha generado el evento.

REMOVEEVENTLISTENER

Al igual que podemos añadir manejadores de eventos con addEventListener, podemos eliminarlos con removeEventListener, cuya sintaxis básica es la siguiente:

```
nodoObjetivo.removeEventListener ("nombreDeEventoSinPrefijo", funcionAEjecutar,
parametroBooleanoOpcional)
```

Donde los parámetros tienen el mismo signficado que con addEventListener.

Un ejemplo nos aclarará las ideas. En este caso, vamos a plantear una variante del código de ejemplo que vimos para addEventListener. Ese código daba lugar a que cuando el usuario pasaba el ratón sobre un elemento de tipo título <h1>, <h2>, … el texto tomara color naranja, y cuando dejara de estar encima de ese elemento tomara color marrón. Pero al repetirse el paso por un elemento que estaba en color marrón, volvía a ponerse naranja porque los manejadores de eventos seguían actuando de la misma manera. Una forma de anular ese cambio de color es eliminar los manejadores de eventos después de que el texto haya cambiado a color marrón. Al anular los manejadores de eventos, una vez el texto toma el color marrón, quedará ya en ese color definitivamente. El cambio a introducir es en la función cambiarColor2, dejando el código como sigue:

```
function cambiarColor2() { this.style.color = 'brown';
this.removeEventListener("mouseover", cambiarColor1);
this.removeEventListener("mouseout", cambiarColor2);
}
```

ATTACHEVENT() Y DETACHEVENT()

Estos métodos fueron usados en el pasado por algunos navegadores como equivalentes a addEventListener y removeEventListener. Desde el momento en que todos los navegadores modernos soportan los métodos addEventListener y removeEventListener, estos métodos no deben ser usados excepto en casos muy concretos en que tengamos que resolver situaciones muy específicas.

PROPAGACIÓN DE EVENTOS. BUBBLING.

Ya hemos indicado que ciertas acciones del usuario generan varios eventos simultáneos. Por ejemplo si tenemos una imagen dentro de un div, y tenemos definida la captura del evento click para ambos, cuando se hace click sobre la imagen se producen en realidad dos eventos: el evento click sobre la imagen y el evento click sobre el div.

La propagación se dice que consta de varias fases: la captura sobre los elementos que sufren el evento detectada por el navegador (que va desde fuera del DOM hacia dentro), el envío del evento al manejador del evento (donde el evento es enviado como un objeto), y el burbujeo, que da lugar a la ejecución del código de respuesta asociado al evento (que va desde dentro del DOM hacia fuera).

El comportamiento que secuencia la ejecución del código de respuesta con un orden determinado se llama “bubble” o “burbujeo”, de modo que desde el elemento más interior del DOM se va ejecutando la respuesta al evento y seguidamente se va expandiendo la ejecución hacia los elementos más externos. En nuestro ejemplo, se ejecutaría en primer lugar la respuesta al evento sobre la imagen por estar dentro del div, y a continuación la respuesta el evento sobre el div. El burbujeo continua sobre todos los elementos del DOM hasta llegar a document y al objeto global window.

Casi todos los eventos burbujean, pero algunos eventos no lo hacen. Por ejemplo el evento focus (onfocus, obtener el foco un campo input) no lo hace. Algunos eventos tienen un burbujeo especial, por ejemplo el evento load (onload, carga de un elemento) burbujea hasta el elemento document, pero no alcanza al elemento window.

¿QUÉ FORMA DE INTRODUCIR MANEJADORES DE EVENTOS ES MEJOR?

Hemos visto tres maneras diferentes de introducir manejadores de eventos: en línea dentro del código HTML, como propiedades de nodos del DOM y a través de la función addEventListener. ¿Cuál es mejor?

Haremos las siguientes consideraciones:

- Introducir los manejadores de eventos en línea es simple pero tiene inconvenientes serios: no separa la estructura de la página web (HTML) del código JavaScript. Esto dificulta el mantenimiento del código y se considera una práctica no adecuada. Por otro lado, se generan conflictos de nombres.

- Introducir manejadores de eventos como propiedades puede resultar adecuado, pero cuando hay que introducir varios manejadores para un mismo evento puede "oscurecer" (hacer más difícil de leer) el código. Por ejemplo para introducir dos manejadores para el evento onclick escribiríamos element.onclick = function () {dispararA(); dispararB()}.

- Introducir manejadores de eventos a través de addEventListener, a pesar de que ha tenido algunos problemas de compatibilidad en el pasado, es un modo que se está estandarizando y se considera también adecuado.

EJERCICIO

Crea un documento HTML donde dentro del elemento body tengamos un div con id "principal", dentro de principal otro div denominado "secundario", y dentro de secundario otro div con id "terciario". Dentro de terciario debe existir un párrafo con el texto: Ejemplo de bubbling (burbujeo). Añade eventListeners con el evento click para los párrafos y todos los elementos div, document y window, y una función de respuesta común para todos ellos que emita el mensaje de alerta <<Soy un nodo tipo NombreDelNodo y estoy burbujeando>>.

Ejemplo: al hacer click sobre el texto <<Ejemplo de bubbling (burbujeo)>> deberán empezar a aparecernos mensajes como: Soy un nodo tipo P y estoy burbujenado. Soy un nodo tipo DIV y estoy burbujeando. Soy un nodo tipo DIV y estoy burbujeando...

Para comprobar si tus respuestas y código son correctos puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU01159E

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=78&Itemid=206